Eight National Conference on
Air Breathing Engines and Aerospace Propulsion
[NCABE –2006]
DIAT Pune
December 12 –14, 2006

PARALLELIZATION OF A TURBOMACHINERY FLOW CODE ON NAL'S FLOSOLVER PARALLEL COMPUTER

R.Prathapanayaka*, T.N.Venkatesh[#], T.R.Shembharkar*

*Scientist, Propulsion Division, NAL, Bangalore – 560 017
[#]Scientist, Flosolver Division, NAL, Bangalore – 560 017
nalprathap@yahoo.com, tnv@flosolver.nal.res.in, trshembharkar@prop.cmmacs.ernet.in

ABSTRACT

A 3-D CFD flow code for multi-stage turbomachinery flow computation, namely the Dawes code 'un_b3d_ke' has been parallelized on the NAL Flosolver parallel computer Mk5 using MPI libraries. The parallelization has been effected employing very basic MPI subroutines. The parallel code performance has been tested for a high pressure ratio fan stage and a 3-stage compressor flow simulations and the results have been compared with the sequential code results. It is found that the time taken by the parallel code for the computation is substantially reduced.
KEY WORDS: TURBOMACHINERY CFD, PARALLEL COMPUTERS, PARALLELIZATION.

1. INTRODUCTION

Complex 3-dimensional flow simulation in multi-stage turbomachines continues to be a time consuming exercise as a large number of grid points are generally required for accurate flow resolution. There is always a need to reduce the turn-over time especially when such flow simulations are required in large numbers in order to investigate the complete performance characteristics or optimize a new blade design. One of the possible methods to reduce the computational time of an existing CFD code, without modifying the solution algorithm, is to parallelize the code. The Propulsion Division, NAL has been using the Dawes code 'un_b3d_ke' for simulating multi-stage turbomachinery flows for quite some time [1-3]. It is a 3-dimensional Navier-Stokes flow solver based on structured H-mesh. It employs the k-ε turbulence model to simulate the turbulent flow. The salient features of the in-built numerical algorithm are: cell-centered finite volume formulation, central differencing for fluxes, artificial viscosity for stability and fourth order Runge-Kutta scheme for time marching. This sequential code has been parallelized recently on a NAL Flosolver parallel computer Mk5. The Flosolver Mk5 machine is built by the Flosolver Division, NAL and has 32 processors. This machine is a distributed memory system [4]; each processor is having 1GB memory. MPI (Message Passing Interface) libraries [5] are loaded on this machine for message passing.

The present paper describes some relevant details about the methodology of parallelization and discusses the performance of the parallel code vis-à-vis the sequential code with regard to two test cases; one a high pressure ratio fan stage and second a 3-stage axial flow compressor.

2. TURBOMACHINERY CFD CODE 'UN_B3D_KE'

The CFD code 'un_b3d_ke' is a 3-dimensional Navier-Stokes flow solver specifically written for turbomachinery flows. It is based on cell centered finite volume formulation with structured H-mesh in cylindrical-polar (r-θ-x) coordinate system. It solves the conservation equations in a rotating frame of reference with absolute velocity components as dependent variables. It employs the k-ε turbulence model to simulate the turbulent flow. It uses central differencing for fluxes with artificial viscosity for numerical stability and fourth order Runge-Kutta scheme for time marching. The code has been written in FORTRAN 77 language.

2.1 IMPORTANT SUBROUTINES OF THE PROGRAM

The original sequential code has 15 subroutines, out of which the following subroutines are important for the purpose of parallelization.

1. MESHGEN: grid generation.
2. SOLVER: 4^{th} order Runge-Kutta time integration, central differencing for fluxes.
3. AREAS: volume and face areas of the cell calculation.
4. AV: artificial viscosity terms calculation.
5. NS: viscous stress terms calculation.
6. DELTAT: time step calculation.
7. DELTAU: calculation of changes in variables due to time step.
8. MODFLX: Modification of fluxes at solid boundaries.

2.2 TIME TAKEN BY THE SUBROUTINES

In order to profile the sequential code on the basis of CPU/system time requirement, a representative flow computation in a single stage axial flow compressor was carried out for a fixed number of iterations. The relative CPU/system time (in percentage) taken by major subroutines are tabulated below (Table-1). The MESHGEN and AREAS subroutines are called once and require little time, hence they may be left unmodified. The subroutines SOLVER, AV, NS, DELTAT, MODFLX and DELTAU are good candidates for parallelization.

Table – 1 **Time taken by some of the subroutines**

No	Name of subroutine	Time taken in %age
1	MESHGEN	00.08
2	**SOLVER**	**36.00**
3	AREAS	00.03
4	**AV**	**13.31**
5	**NS**	**08.35**
6	DELTAT	02.56
7	**MODFLX**	**10.34**
8	**DELTAU**	**23.35**
9	Other sub routines	05.98

3. FLOSOLVER Mk5

Flosolver Mk5 machine has been built by the Flosolver Division, NAL. The main features of this machine are as follows,

1. 32 processors
2. Intel Pentium III@ 450 MHz speed
3. 1GB RAM / Processor
4. Connectivity: NAL flow switch & Ethernet.
5. Linux operating system
6. Loaded with MPI libraries
7. Distributed memory system [4].

As this machine contains 32 processors, the parallel version of the 'un_b3d_ke' code is expected to be of great use in terms of time saving. The machine is of distributed memory (each processor has memory which is accessible only by itself) type and parallelization of the sequential FORTRAN code is done using MPI libraries.

4. GENERAL PRINCIPLES OF PARALLELIZATION

The general steps required for parallelization of a sequential code can be broadly listed as follows [4]:

1. Finding parallelism in the sequential code.
2. Profiling the code in terms of CPU/system time requirement of each part of the sequential code.
3. Dependency study to identify parts of the code, which are independent/dependent of each other.
4. Division of the task such that each independent task could be assigned to different processing unit.
5. Identification of variables to be communicated among different tasks.
6. Sequential code modification using the MPI library.

The parallelization is based on the domain decomposition method. Here the computational task is subdivided into subtasks for different processors by dividing the domain based on number of processors. The implementation uses single process multiple data (SPMD) model of programming where the same source code is compiled and executed in each processor. The necessary communication and exchange of data between the processors is ensured by the MPI library routines.

5. PARALLELIZATION OF 'un_b3d_ke'

The process of parallelization of an existing sequential code on a given machine is strongly determined by the algorithm and structure of the code. A CFD code like the one under consideration, namely 'un_b3d_ke' which is based on an explicit formulation is intrinsically suited to efficient parallelization.

The code 'un_b3d_ke' is a 3-dimensional multi-stage turbomachinery CFD code. A typical computational domain for turbomachinery problem is shown in Figure 1. Its 3 dimensions are identified by blade-to-blade or circumferential (I-index), inlet-to-outlet or axial (J-index) and hub-to-casing or radial (K-index) directions. In a general axial turbomachinery CFD calculation, the number of grid points will be more in axial (J) direction because of multiple blade rows. Therefore, it is more meaningful to divide the computational domain in axial direction. In the code 'un_b3d_ke', the axial index J (for cells) starts from 1 and ends at JMM1. By dividing the axial index JMM1 by the number of processors, the length of the subtask for each processor has been calculated. The other two directions (I & K) have been kept same as the sequential code. As the code employs 5-point computational stencil (in one direction) for certain quantities (artificial viscosity terms), it needs at least J-2, J-1, J, J+1, and J+2 data values. The subtask assigned to each processor is, therefore, not completely independent data-wise but requires data from an adjacent processor. Actually, a particular processor must receive 2 rows (2-dimensional planes) of data J+1 and J+2 from the next processor and must send 2 rows of data J-1 and J-2 to it. Figure 2 shows this inter-dependence of data schematically. It is this communication of data, which is accomplished by introducing MPI library in the sequential code.

Message Passing Interface (MPI) library has more than 115 routines for creating and maintaining MPI environment and facilitating message passing. We have used a limited number of basic MPI routines and also combined them according to the requirement in separate user-defined subroutines to be called in the code.

To start the process of parallelization, a MPI environment is created right in the MAIN of the code. As noted above, the subroutines MESHGEN and AREA do not need parallelization as they are called only once and take relatively little time for execution. Therefore, each processor executes these subroutines and has geometrical information about the grid in its own memory. The subroutines SOLVER, AV, NS, DELTAT, MODFLX and DELTAU are good candidates for parallelization. To parallelize these subroutines, we need to identify all the do-loops on J index and change the beginning and ending J index values to corresponding values of the domain assigned to each processor. All the processors thus perform the computations in parallel. But during this process, a processor (say, 1) needs some data (J+1, J+2) from adjacent processor (say, 2) and processor 2 needs some data (J-1, J-2) from processor 1 for carrying out the computations. In an explicit code like 'un_b3d_ke', this needed data is from previous sweep (each intermediate step of R-K method loop) and therefore can be transferred from neighboring processors before this sweep starts. All the processors can then carry out computation in parallel independent of each other. This is the strategy that has been followed here.

The data transfer has been done by introducing separately written subroutine IPCS which utilizes the very basic point-to-point communication routines MPI_SEND and MPI_RECV, and collective communication routine MPI_BCAST. It transfers the data (2 J planes) to an adjacent processor. The whole parallelization of the code has been accomplished with the help of these basic MPI library routines.

6. RESULTS & DISCUSSION

Two compressors, a single stage high pressure ratio fan [6] and a 3-stage compressor were tested to check the performance and validity of the parallel version of the 'un_b3d_ke' code in 1, 2, 4 and 8 processors mode and results were validated with the sequential version of the code. Due to constrain on the length of the paper, we will restrict our discussion to the first case, namely the fan stage. The high pressure ratio fan stage (NAL-CAE) was designed and developed under a joint programme between Chinese Aeronautical Establishment (CAE), China and National Aerospace Laboratories (NAL), Bangalore, India [6]. This fan is a single stage axial flow compressor with a pressure ratio of 2.0.

Flow computations were carried out with both sequential and parallel versions of the code 'un_b3d_ke'. The grid for the complete fan stage was 33 (radial, index K) x 376(axial, index J) x 33 (tangential, index I) as shown in Figure 1. The boundary conditions for the computations included total pressure, total temperature, and flow angles at inlet, static pressure at exit tip with radial equilibrium condition, no slip condition at walls and periodic condition in tangential direction. The frozen rotor model was used to simulate the interface between the rotor and the stator. The solutions for different back pressures were obtained starting from the nearest lower exit static pressure solution. Complete performance curves at the design speed of 22400 rpm for the fan were obtained.

Figures 4 shows the performance map obtained from the sequential code and the parallel code (with 8 processors) respectively for the high pressure ratio fan. There is total agreement in the two results. Similarly the two Mach number distributions at the same location shown in Figure 3 are similar to each other. These results confirm the correctness of the parallelization process and validate the parallel version of the code. The performance of the parallel code can be ascertained from the following two figures. Figures 5 shows the speed-up due to number of processors with reference to the single processor. It indicates the computational time required for convergence has reduced considerably. The speed-up with 2 processors is 1.875, with 4 processors 3.173 and with 8 processors 4.94. Ideally, the speed-up should be linear but in practice it is not obtained due to increased data transfer overhead in the parallel code. The corresponding parallelization efficiency values for the parallel code are 93.75%, 79.33% and 61.75%.

Similar comparative exercise was carried out with the 3-stage axial flow compressor. Computations were carried out with 33x331x33 grid size. The speed-up for the parallel code with 2 processors is 1.846,

with 4 processors 3.1566 and with 8 processors 4.584. The corresponding efficiency values are 92.32, 78.91 and 57.3 respectively in this case.

7. CONCLUSIONS

Turbomachinery CFD code 'un_b3d_ke' is being used at Propulsion Division, NAL for quite some time. This code has been parallelized successfully to get the advantage of reduction in computation time. Parallelization of the code 'un_b3d_ke' has indeed been a rewarding exercise. The parallelization has been done using domain decomposition technique and standard MPI libraries have been used for message passing. The domain decomposition is done based on number of processors. This makes the parallelization general and this parallelized code can be used in any standard parallel computer.

The parallel version has been tested for two compressors; a single stage compressor and a multi-stage compressor. The performance of the parallelization is satisfactory, as time taken for obtaining solution has come down substantially. Using this parallelized code on Flosolver Mk5 machine, complex multi-stage turbomachinery flow computations can be successfully carried out within a decent practical time frame (say, within a day) by using 4 to 8 processors. The parallelization has not affected the accuracy of the solution obtained by the code. Parallel version results agree very well with the sequential code results for both single stage and multi-stage compressors.

It is however observed that the efficiency of the parallel code comes down with higher number of processors even though the computation time continuously decreases. The best choice of number of processors for a particular problem may be a compromise between desired speedup and efficiency.

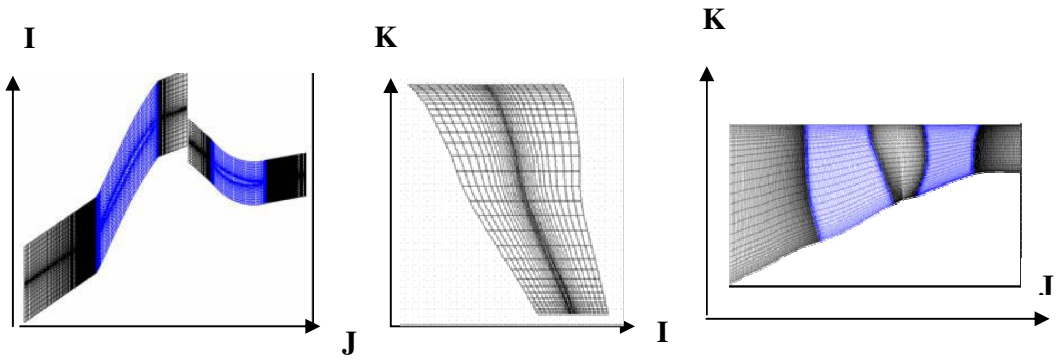


Fig. 1. Typical grid and grid indices for flow computation

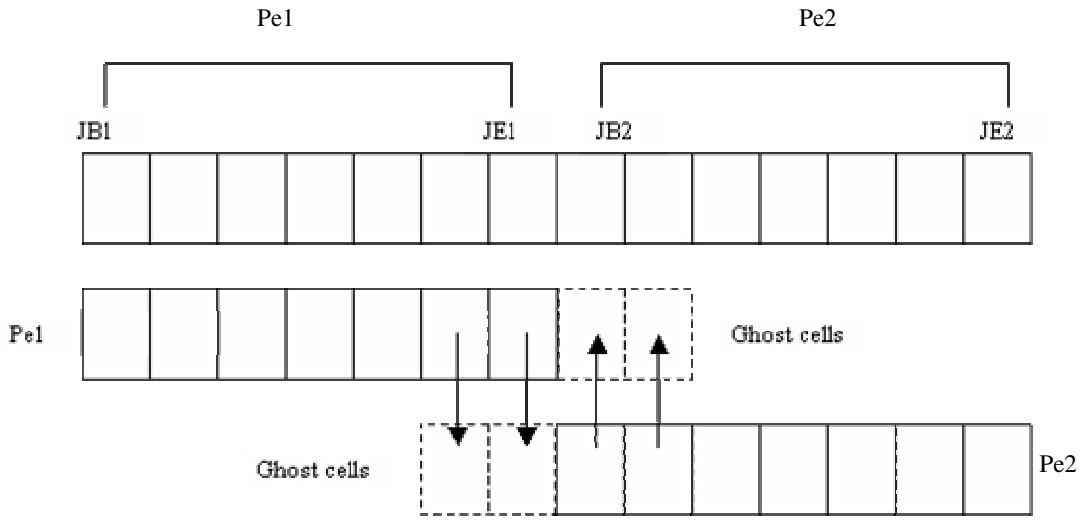


Fig. 2. Communication between processors.
(Pe-Processor, JB- Beginning J, JE-Ending J)

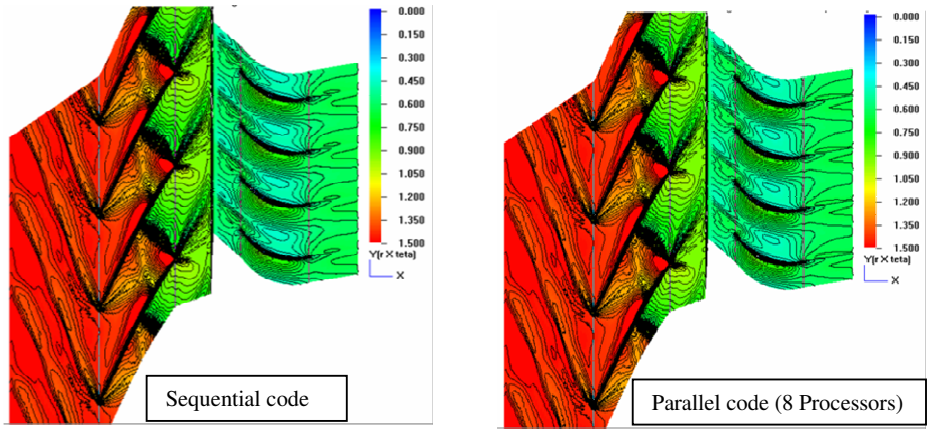
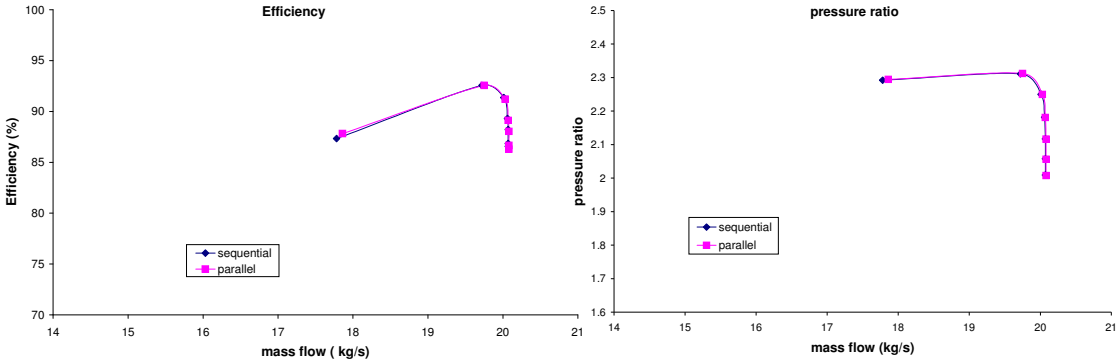


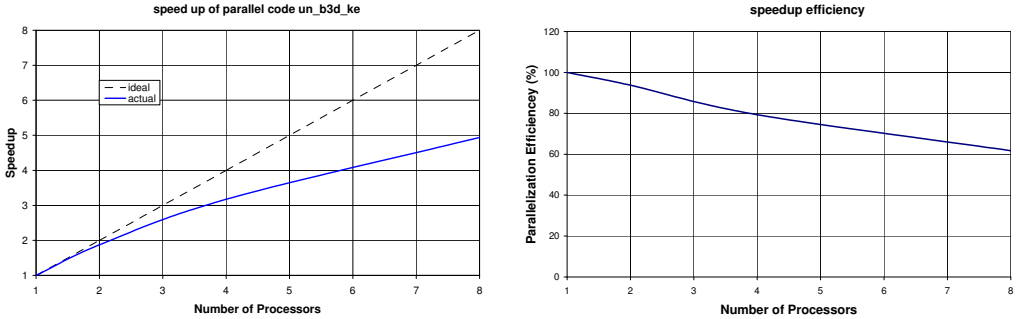
Fig. 3. Mach number contour on a K-plane (casing)



(a) Pressure ratio Vs. Mass flow (kg/s)

(b) Efficiency (%) Vs. Mass flow (kg/s)

Fig. 4. Performance maps of the single stage fan



(a) Speed-up Vs. Number of processors

(b) Parallelization Efficiency Vs. Number of processors

Fig. 5. Performance of the parallel code (Single stage fan computation)

REFERENCES:

1. R.Prathapanayaka, T.R.Shembharkar, "CFD analysis of Kaveri 3-stage fan and performance evaluation", NAL Project Document PD-PR-0210, December 2002.
2. S.Sangamnath, T.R.Shembharkar, "CFD flow analysis of axial flow compressor stage with different tip clearances", NAL Project Document PD-PR-0309, October 2003.
3. S.Sangamnath, T.R.Shembharkar, "CFD flow analysis of NAL-CAE fan stage with re-stacked rotor blade", NAL Project Document PD-PR-0310, October 2003.
4. T.N.Venkatesh, "Parallel programming using MPI/Shared memory", High Performance Computing, NAL-UNI Lecture Series #16, December 2000.
5. MPI-forum, http://www.mpi-forum.org
6. Chen, M.Z., Xu Liping, M.V.A.Murthy, M.Jayaraman, B.R.Pai, R.Prathapanayaka, 2001, 'Development of an advanced high pressure ratio transonic fan stage, Part – I: Design and analysis', Proceedings of Fifteenth International Symposium on Air Breathing Engines, Bangalore, India.